MASTER DEGREE in EMBEDDED COMPUTING SYSTEMS
A.Y. 2017 - 2018

Schedulability Analysis Tool for Hierarchical Real Time Components

Component Based Software Design

Professors
Tommaso Cucinotta
Luca Abeni
Marco Di Natale
Mauro Marinoni

Student
Silvio Bacci

Index

1 Introduction

1.1 Introduction

The objective of the project was to extend the schedulability analysis tool for hierarchical real-time EDF and FP components developed by the Professor Luca Abeni. The extension aims to add to the tool an analysis in case of EDF and multi-CPU partitioned scheduling. In addition the tool was also equipped with an useful research tool used to find a periodic server able to schedule a specific taskset.

1.2 Project requirements

The original project requirements are reported hereafter:

"Extend Abeni's schedulability analysis tool for hierarchical RT EDF/FP components to EDF and Multi-CPU partitioned case"

In the following chapter we are going to explain all the theoretical results regarding:

- Schedulability analysis for periodic tasks
- Schedulability analysis for Hierarchical components
- Schedulability analysis for multi-CPU partitioned case

2 Schedulability Analysis for Periodic Tasks

2.1 Introduction

In this chapter we are going to analyse the main theoretical results useful to conduct a schedulability analysis on a given taskset. In particular we are going to explain how we used these results in order to realize the tool. Let us consider separately the case of fixed and dynamic priority analysing the different results in case of constrained deadline.

2.2 Real-time concepts

First of all we need to introduce some some formal definition used below. Let us define a taskset T as a set of tasks τ_i , where each task can be expressed according to the **(C, D, T) model** as:

$$\tau_i = (C_i, D_i, T_i)$$

where C_i is the computational time, D_i is the deadline and T_i is the period. For what concerns the scheduling algorithms we are going to consider just the following cases:

- **FP**
  - Fixed priority algorithm in the general sense (task order chosen off-line by the user)
- **RM**
  - Fixed priority algorithm in which the tasks are ordered by increasing periods
- **DM**
  - Fixed priority algorithm in which the tasks are ordered by increasing deadlines
- **EDF**
  - Dynamic priority algorithm

2.3 Taskset with deadlines equal to periods

In many cases it is useful to have a very simple test to see if the taskset is schedulable. In particular we are going to explain the main approaches and tools used to check schedulability.

2.3.1 Utilization bound analysis

A sufficient (and necessary in case of EDF) test is based on the utilization bound. The utilization least upper bound (U_{lub}) for a scheduling algorithm A is the smallest possible utilization U_{lub} such that, for any taskset T , if the taskset's utilization U is not greater than U_{lub} (i.e. $U \leq U_{lub}$), then the task set is schedulable by algorithm A . Therefore the schedulability test consist in computing the taskset's utilization as:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

Then if $U \leq U_{lub}$ the taskset is schedulable, if $U > 1$ the taskset is not schedulable while if $U_{lub} < U \leq 1$, the taskset may or may not be schedulable. In general no scheduling algorithm can schedule a task set if $U > 1$, so U_{lub} must be less than or equal to one and in particular an algorithm A is optimal if its $U_{lub} = 1$. Let's try to find out the different values for U_{lub} according to different scheduling algorithms.

Theorem (Liu and Layland, 1973):

Given a task set of periodic (or sporadic) tasks, with relative deadlines equal to periods, the task set is schedulable under RM if

$$U \leq U_{lub} = n \cdot (2^{1/n} - 1)$$

□

If we consider the a taskset of periodic (or sporadic) tasks, with relative deadlines less than periods, we can define another taskset's utilization that is:

$$U' = \sum_{i=1}^{n} \frac{C_i}{D_i}$$

Then the test is the same as the one for RM (or DM), except that we must use U' instead of U.

Theorem:

Given a task set of periodic or sporadic tasks, with relative deadlines equal to periods, the task set is schedulable by EDF if and only if

$$U \leq U_{lub} = 1$$

□

Note that in this case we have a necessary and sufficient condition. This means that for what concerns the fixed priority algorithms we need something more to achieve a necessary condition.

2.3.2 Workload analysis

As we have already seen, we have not a sufficient and necessary condition in case of fixed priority scheduling algorithms in general. The basic idea is to introduce a sufficient and necessary condition based on the workload of the taskset. First of all let us introduce the workload function that is:

$$W_i(t) = C_i + \sum_{j<i} \lceil \frac{t}{T_j} \rceil C_j$$

Theorem (Lehoczky, Sha and Ding, 1989):

A set of fully preemptive periodic tasks can be scheduled by a fixed priority algorithm if and only if

$$\forall i = 1, \ldots, n \qquad \exists t \in (0, D_i] \qquad : \qquad W_i(t) \leq t$$

□

Later, Bini and Buttazzo restricted the number of points in which the condition has to be checked to the following testing set:

$$TS_i = P_{i-1}(D_i)$$

where $P_i(t)$ is defined by the following recurrent expression:

$$\begin{cases} P_0(t) = t \\ P_i(t) = P_{i-1}(\lfloor \frac{t}{T_i} \rfloor T_i) \cup P_{i-1}(t) \end{cases}$$

Theorem (Bini and Buttazzo, 2004):

A set of fully preemptive periodic tasks can be scheduled by a fixed priority algorithm if and only if

$$\forall i = 1, \dots, n \qquad \exists t \in TS_i \qquad : \qquad W_i(t) \leq t$$

□

2.3 Taskset with deadlines less than periods

In this case the workload analysis is still valid and so we have already a necessary and sufficient condition for fixed priority algorithms. The problem in this case regards EDF because we have no more an useful utilization bound test.

2.3.1 Processor demand approach

For what concerns EDF we need to introduce a new necessary and sufficient test called processor demand approach. First of all let us introduce the demand bound function that is:

$$dbf(t) = \sum_{i=1}^{n} \lfloor \frac{t + T_i - D_i}{T_i} \rfloor C_i$$

Theorem:

A set of synchronous periodic tasks with relative deadlines less than or equal to the periods can be scheduled by EDF if and only if U < 1 and

$$\forall t \in D \qquad dbf(t) \leq t$$

where

$$D = \{ d_k \mid d_k \leq min[H, max(D_{max}, L^*)] \}$$

and

$$L^* = \frac{\sum_{i=1}^{n} (T_i - D_i) U_i}{1 - U}$$

□

As we can see this approach can be applied both in case constrained and non-constrained deadlines. So we will use the processor demand approach to test the schedulability of a taskset in both the cases (as we do with the workload analysis for fixed priority algorithms).

2.4 Summary

Just for recap, let us consider the following table:

	D = T	**D < T**
Fixed Priority	if: utilization bound iff: workload analysis	iff: workload analysis
EDF	iff: utilization bound / processor demand approach	iff: processor demand approach

2.5 Implementation

In the final implementation the schedulability of a taskset under fixed priority is checked using the workload analysis while under EDF using the processor demand approach. In particular we are going to explain the libraries and source files created. The first important library introduced is called "*dbf.c/h*" in which we can find the functions able to compute the workload and processor demand functions (the name of the file will be better explained in the next chapter). The second and more important library is called "*s_analysis.c/h*". Inside this file we can find two important functions: *s_analysis_fp* and *s_analysis_edf*. These functions are the simply a C implementation of the workload analysis (*s_analysis_fp*) and the processor demand approach (*s_analysis_edf*). Finally we can find also a source file called "*s_analysis.c*" from which we call the previous library to check the schedulability of a taskset specified in a given file.

3 Schedulability Analysis for Hierarchical Components

3.1 Introduction

In this chapter we are going to analyse the main theoretical results useful to conduct a schedulability analysis on a given component. First of all we need to introduce some concepts about scheduling of components. In the general case an application is composed by a set of components within which we can find a set of tasks (i.e. taskset). The main idea is to use a two-level hierarchical scheduling system, in which we have two schedulers:

- Host (or global, root) scheduler for scheduling components
- Local (or 2nd level) scheduler for scheduling tasks inside each component

This can be easily done considering components isolated thanks to different VMs. This means that the global scheduler will schedule just VMs (or components) while the local scheduler will be the scheduler running inside the VM.

3.2 Demanded and supplied time

Simplifying a little bit the problem, we can test the schedulability of a component knowing the time needed by the component to respect its temporal constraints and the amount of time provided by the global scheduler. In fact a component will be schedulable if:

$$\text{demanded time} \quad \leq \quad \text{supplied time}$$

where the demanded time is the amount of time (in a time interval) needed by a component while the supplied time is the amount of time (in a time interval) given by the global scheduler to a component.

3.2.1 Demanded time

The good new in this case is that the demanded time needed by the component can be easily computed using the processor demand function (for EDF) and the workload function (for FP).

$$dbf(t) = \sum_{i=1}^{n} \lfloor \frac{t + T_i - D_i}{T_i} \rfloor C_i \quad \textit{(EDF)}$$

$$W_i(t) = C_i + \sum_{j<i} \lceil \frac{t}{T_j} \rceil C_j \quad \textit{(RM)}$$

3.2.2 Supplied time

For what concerns the supplied time we will consider the case of a periodic server. This means that the idea is to use a periodic server to schedule the components (e.g. the global scheduler could be a constant bandwidth server). Remembering that a periodic server is described by the budget Q_s and by the period T_s, the supplied bound function can be expressed as:

$$sbf(t) = \begin{cases} 0 & if \quad t < 2\cdot(T_s - Q_s) \\ (n-1)\cdot Q_s & if \quad n\cdot T_s - Q_s \le t < (n+1)\cdot T_s - 2\cdot Q_s \\ t + n\cdot Q_s - (n-1)\cdot T_s & if \quad (n+1)\cdot T_s - 2\cdot Q_s \le t < (n+1)\cdot T_s - Q_s \end{cases}$$

3.3 Schedulability analysis

Theorem (fixed priority):

A component (i.e. a set of periodic tasks with relative deadlines less than or equal to periods) characterized by a fixed priority local scheduler is schedulable by the global scheduler if and only if

$$\exists t \in TS_i \cup SBF_i \quad : \quad dbf(t) \le sbf(t)$$

where TS_i is the testing set introduced during the workload analysis explanation (chapter 2) and SBF_i is the testing set given by all the points in which the supplied bound function starts to become flat, i.e.:

$$SBF_i = \{ t \mid t = n\cdot Ts - Qs, n \ge 2, t < max\{TS_i\} \}$$

where Q_s and T_s are respectively the budget and the period of the periodic server.

□

Theorem (EDF):

A component (i.e. a set of periodic tasks with relative deadlines less than or equal to periods) characterized by a EDF local scheduler is schedulable by the global scheduler if and only if:

$$\forall t \in D \cup SBF \quad : \quad dbf(t) \leq sbf(t)$$

where D *is the testing set introduced during the processor demand approach explanation (chapter 2) and* SBF *is the testing set given by all the points in which the supplied bound function starts to become flat, i.e.:*

$$SBF = \{ t \mid t = n \cdot Ts - Qs, n \geq 2, t < max\{D\} \}$$

where Q_s *and* T_s *are respectively the budget and the period of the periodic server.*

□

3.4 Implementation

In the final implementation the schedulability of a taskset is checked using the result of the last two theorems. In particular we are going to explain the libraries and source files created. The first important library introduced is called "*sbf.c/h*" in which we can find the function able to compute the supplied bound function for a given periodic server. The second library is called "*h_analysis.c/h*". Inside this file we can find two important functions: *h_analysis_fp* and *h_analysis_edf*. These functions are the simply a C implementation of the last two theorems. In particular the idea is to use the functions contained in the "*dbf.c/h*" library (described in the previous chapter) to compute the workload and processor demand functions and compare them to the specified periodic server. Finally we can find also a source file called "*h_analysis.c*" from which we call the previous library to check the schedulability of a taskset specified in a given file.

4 Search Periodic Server

4.1 Introduction

In this chapter we are going to explain how we can find a specific periodic server able to schedule a given taskset. Of course the objective in this case is to find the parameters of the server, i.e. budget Qs and period Ts.

4.2 Algorithm

The idea behind the periodic server research is to try a subset if all the possible periodic servers and take the best one. Of course we need to specify what means best in this case. In general the tool aims to find periodic server, able to schedule a taskset, with the minimum bandwidth. This means that we want to find the smallest periodic server that can solve our problem.
In order to performing this research we test the server changing the parameters. In particular the ranges used are:

- $Ts \in [min\{T_i\}, 2 \cdot max\{T_i\}]$
- $Qs \in [Ts \cdot U, Ts \cdot BW]$ where BW is the current best bandwidth and U is the utilization of the taskset

The budget's interval is restricted to the presented values considering that a periodic server needs a bandwidth at least equal to the utilization of the taskset (otherwise the server cannot schedule it) and that it is useless to check values of the budget for which the budget gets worse than the current best one. In general every time we change the parameters we take note of the best periodic server, i.e. the one with the minimum bandwidth.

4.3 Implementation

In the final implementation we can find a function called *find_periodic_server* in the library "*h_analysis.c/h*". This function performs a two nested for cycles in which we test all the period and budget as explained above. Finally we can find also a source file called "*find_ps.c*" from which we call the previous library to find a periodic server for a taskset specified in a given file.

5 Schedulability Analysis for M-CPU partitioned scheduling

5.1 Introduction

In this chapter we are going to analyse the main theoretical results useful to conduct a schedulability analysis on multi-CPU machine. First of all we need to introduce some concepts about the possible scheduling approaches in case of multi-processor machines. In general we can can have two approaches:

- Partitioned scheduling
- Global scheduling

In the global scheduling case (the one we are going to ignore in this project), the idea is to use a single task queue shared by M CPUs and select the first M ready tasks. On the other side in the partitioned scheduling the idea is to reduce the problem assigning statically tasks to M CPUs and then performing uniprocessor scheduling for each CPU. The good new using the partitioned scheduling is that we can use all the tools presented in the previous chapters to check the schedulability of the tasks.

5.2 Bin-packing

The real problem with partitioned scheduling is to find an optimal way to assign tasks to the different cpus. In fact this problem, also known as bin-packing, is a NP-hard problem in the general sense and so we have not an optimal solution. However, in the literature we can find lots of different heuristic bin-packing algorithms useful for our purposes:

- **Best fit**
  - Assign each task to the processor with the smallest empty space.
- **First fit**
  - Assign each task to the first processor that can schedule it.
- **Next fit**
  - Assign each task to the same processor used for the previous task. If the new task can be scheduled together with the tasks already present continue, otherwise assign the task to a next empty cpu.
- **Worst fit**
  - Assign each task to the processor with the largest empty space, otherwise assign the task to a next empty cpu.
- **First fit decreasing**
  - Sort tasks in decreasing order of utilization, then assign tasks according to the first fit

Let us define M_{on} as the number of bins used by an online algorithm and M_o the optimal number of bins. Any online algorithm uses at least 4/3 times the optimal number of bins:

$$M_{on} \geq \frac{4}{3} M_o$$

Is addition some useful results are:

- NF and WF never use more than $2M_o$ bins.
- FF and BF never use more than $\frac{17}{10} M_o + 1$ bins.
- FFD never uses more than $\frac{11}{9} M_o + 4$ bins.

5.3 Utilization under fixed priority

The heuristic algorithms explained before are useful and widely used for scheduling task under EDF. The problem borns when we deal with fixed priority algorithms. In fact in this case we can use of course once again solutions like "*First fit*" and "*Next fit*" but we cannot use, or at least in this version, the other algorithms. The problem is related to the utilization factor computed as:

$$U = \sum_i \frac{C_i}{T_i}$$

If we think about the previous relation we can understand that it has no sense for fixed priority. So the solution is to use another kind of utilization. Remember that the schedulability check under fixed priority says that we need to verify:

$$\forall i = 1, \ldots, n \qquad \exists t \in (0, D_i] \qquad : \qquad W_i(t) \leq t$$

If we use this result we can compute a new utilization factor as:

$$U_i = \frac{W_i(t)}{t}$$

Moreover the general utilization of the entire taskset will be:

$$U_i = max_\tau \{ min \{ \frac{W_i(t)}{t} \} \}$$

This means that for every task we are searching the minimum possible value (note that some values could be also greater than 1). Then among all the tasks, we are going to take the maximum value that will be always less than or equal to one (if the taskset is schedulable).

At the end we can use this utilization for what concerns the algorithms "*Best fit*" and "*Worst fit*". The only algorithm we cannot use under fixed priority is "*First fit decreasing*" because we cannot order the tasks in a different way from the one chosen fixed.

5.4 Implementation

In the final implementation the schedulability of a taskset in a multi-CPU environment is checked using the schedulability checking tools explained in the previous chapters and the bin packing algorithms. In particular we are going to explain the libraries and source files created. The first important library introduced is called "*mcpu_analysis.c/h*" in which we can find the function able to perform the desired bin packing algorithm. As already said it uses different ways to compute the utilization of the taskset according to the scheduling algorithm (fixed priority of EDF). The schedulability on a specific CPU is checked using the "*h_analysis*" tool if a periodic server is specified for the CPU. Moreover the function is also able to find a periodic server if no one is specified. In this case the schedulability check is perform using the "*s_analysis*" tool. Finally we can find also a source file called "*mcpu_analysis.c*" from which we call the previous libraries to check the schedulability of a taskset on a specific vm, both of them specified in a given file.

6 Implementation

6.1 Introduction

In this chapter we are going to explain how the tool was organized and developed. First of all is important to note that the tool was developed using the C programming language and compiled thanks to the gcc compiler installed on machine running an OS Linux distribution Debian 8 (Jessie). The idea behind the organization of the files is to divide the code in order to obtain reusable components able to works also in completely different systems.

6.2 Project files

The project folder is composed by the following folder:

- bin
- lib
- src
- taskset
  - Contains the files in which are defined the different taskset used during testing
- vm
  - Contains the files in which are defined the different vm used during testing
- report
  - Contains the report of the project

The *bin* folder is used to store the executable files obtained after the compilation and linkage of the source files. The src folder contains the source files able to perform the various schedulability analysis presented. In particular we can find the following programs:

- sbf
  - Input: budget and period of the periodic server
  - Output: compute the supplied bound function in a given interval of time ($t \in [0, 100]$)
- s_analysis
  - Input: taskset and scheduling algorithm
  - Output: check schedulability of the taskset under the specified scheduling algorithm
- h_analysis
  - Input: taskset, budget and period of the periodic server and scheduling algorithm
  - Output: check schedulability of the taskset under the specified scheduling algorithm with the specified periodic server
- find_ps_server
  - Input: taskset and scheduling algorithm
  - Output: search the best periodic server able to schedule the taskset under the specified scheduling algorithm

- mcpu_analysis
  - Input: taskset, vm, scheduling algorithm and allocation algorithm
  - Output: check schedulability of the taskset considering the parameters of the VMs (for all of them run the same specified scheduling algorithm) and the specified allocation algorithm

The *lib* folder is used to store the library (i.e. reusable components) and we can find the following organization:

- task
  - task_io.c/h
  - taskset.c/h
  - sorting.c/h
  - utilities.c/h
  - structs
    - task.h
    - taskset.h
    - periodic_server.h

- vm
  - vm_io.c/h
  - sorting.c/h
  - utilities.c/h
  - structs
    - cpu.h
    - vm.h

- schedulability
  - dbf.c/h
  - sbf.c/h
  - s_analysis.c/h
  - h_analysis.c/h
  - mcpu_analysis.c/h

In the *task* folder we can find all the useful libraries to load and manage the taskset. For example we can find all the functions useful to compute the taskset's utilization, max period, sort taskset according to different policies and so on.
In the *vm* folder we can find the libraries useful to load and manage the vm. In particular we can find all the functions useful to sort the different CPU according to different policies.
In the *schedulability* folder we can find the main libraries of the project. In particular in each file we can find the schedulability analysis as described in the previous chapter.

7 Testing

7.1 Introduction

In this chapter we are going to report all the testing results. In particular we used the tool provided by the professor Abeni to generate random task sets with a desired utilizations and with tasks's periods between 5 and 30. The two task sets used for testing the tool are:

TASK SET ID: 1			
TASK ID	**C**	**D**	**T**
1	2	7	7
2	5	15	15
3	2	7	7

TASK SET ID: 2			
TASK ID	**C**	**D**	**T**
1	2	11	11
2	2	16	16
3	3	14	14
4	3	11	11
5	2	28	28

In addition to the previous one we add also a task set described in [8] for which we know already the exact CPU allocation in the case of multi-processor machine:

TASK SET ID: 3			
TASK ID	**C**	**D**	**T**
1	2	10	10
2	5	10	10
3	4	10	10
4	7	10	10
5	1	10	10
6	3	10	10
7	8	10	10

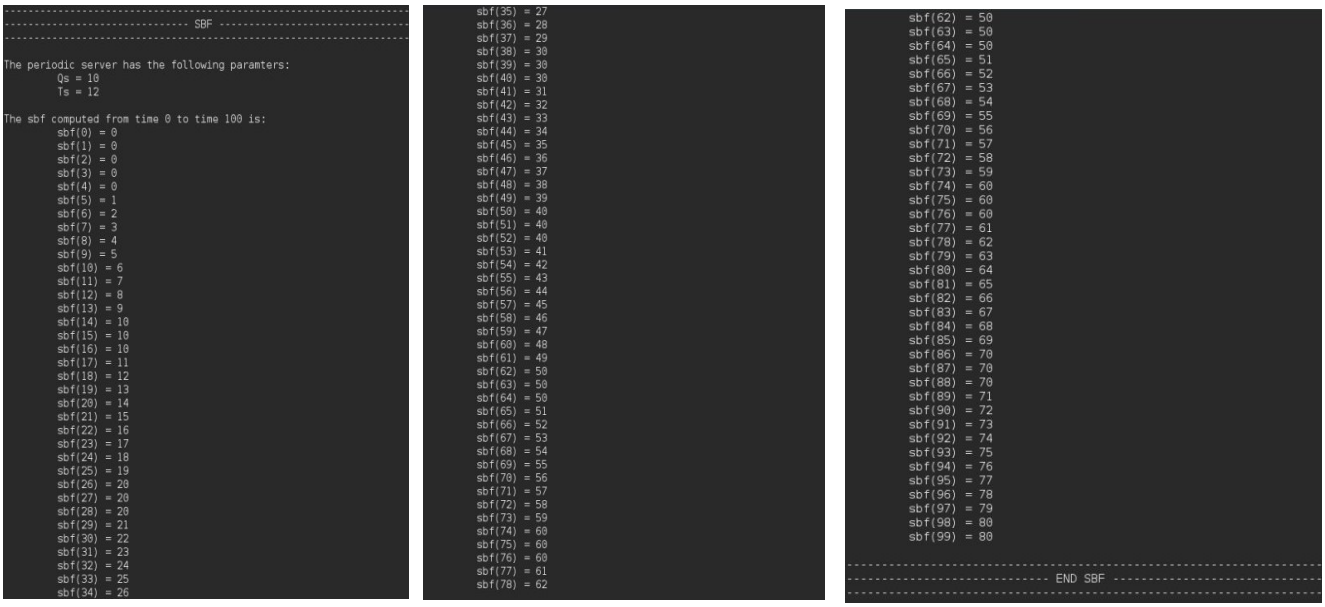
In addition we have created 2 different machines, both of them with 5 processors but the difference is that in one case we defined also the periodic server for each CPU while in the other one the server is found (if it exists) by the tool. So the two machines defined are:

VM ID: 1			
CPU ID	**MAX U**	**Qs**	**Ts**
1	1	10	10
2	1	10	10
3	1	10	10
4	1	10	10
5	1	10	10

VM ID: 2			
CPU ID	**MAX U**	**Qs**	**Ts**
1	1	-	-
2	0,9	-	-
3	0,8	-	-
4	1	-	-
5	0,7	-	-

7.2 SBF

The first thing we check is the correctness of the *sbf* functions that computes the supplied bound function for a specific server from time 0 up to time 100. For example if we compute it for the periodic server with $Qs = 10$ and $Ts = 12$, we obtain:

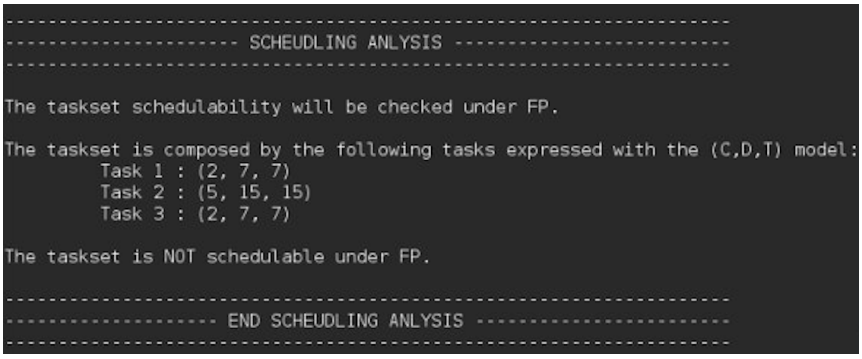


7.3 Schedulability analysis

The second tool we are going to test is the *s_analysis* tool that checks the schedulability of a periodic taskset as it should run on a single processor. Let's apply the tool to the task set 1. Of course we can choose to schedule according 4 different scheduling algorithm, i.e. EDF, RM, DM and a fixed priority in the general sense.

7.3.1 Fixed priority

Let's start by testing the schedulability under fixed priority choosing as priority order the one output by the random task set generator (reported in the table above):



The tool says us that the task set is not scheduled under fixed priority. Let's see if it works properly computing by hand the various calculations:

$$TS_1 = \{7\}$$

$$W_1(7) = C_1 = 2 \leq 7$$

The workload is less than the interval of time and so we can continue.

$$TS_2 = \{7, 14, 15\}$$

$$W_2(7) = C_2 + \lceil \frac{7}{7} \rceil \cdot C_1 = 5 + 2 = 7 \leq 7$$

The workload is less than the interval of time in at least one point and so we can continue.

$$TS_3 = \{7\}$$

$$W_2(7) = C_3 + \lceil \frac{7}{15} \rceil \cdot C_2 + \lceil \frac{7}{7} \rceil \cdot C_1 = 2 + 5 + 2 = 9 > 7$$

The workload is greater than the interval of time of all the scheduling points for task 3 and so it is true, the task set is not schedulable under fixed priority.

7.3.2 RM

Let's see what happens when we choose to use RM as scheduling algorithm:

```
------------------------------------------------------------------
--------------------- SCHEUDLING ANLYSIS -------------------------
------------------------------------------------------------------

The taskset schedulability will be checked under RM.

The taskset is composed by the following tasks expressed with the (C,D,T) model:
        Task 1 : (2, 7, 7)
        Task 3 : (2, 7, 7)
        Task 2 : (5, 15, 15)

The taskset is schedulable under RM.

------------------------------------------------------------------
------------------- END SCHEUDLING ANLYSIS -----------------------
------------------------------------------------------------------
```

The taskset is schedulable and we are going to compute all the calculations once again to check the correctness of the tool:

$$TS_1 = \{7\}$$

$$W_1(7) = C_1 = 2 \leq 7$$

The workload is less than the interval of time and so we can continue.

$$TS_2 = \{7\}$$

$$W_2(7) = C_2 + \lceil \frac{7}{7} \rceil \cdot C_1 = 2 + 2 = 4 \leq 7$$

The workload is less than the interval of time in at least one point and so we can continue.

$$TS_3 = \{7, 14, 15\}$$

$$W_2(7) = C_3 + \lceil \frac{7}{7} \rceil \cdot C_2 + \lceil \frac{7}{7} \rceil \cdot C_1 = 5 + 2 + 2 = 9 > 7$$

$$W_2(14) = C_3 + \lceil \frac{14}{7} \rceil \cdot C_2 + \lceil \frac{14}{7} \rceil \cdot C_1 = 5 + 4 + 4 = 13 \leq 14$$

The workload is less than the interval of time in at least one point and so the task set is schedulable.

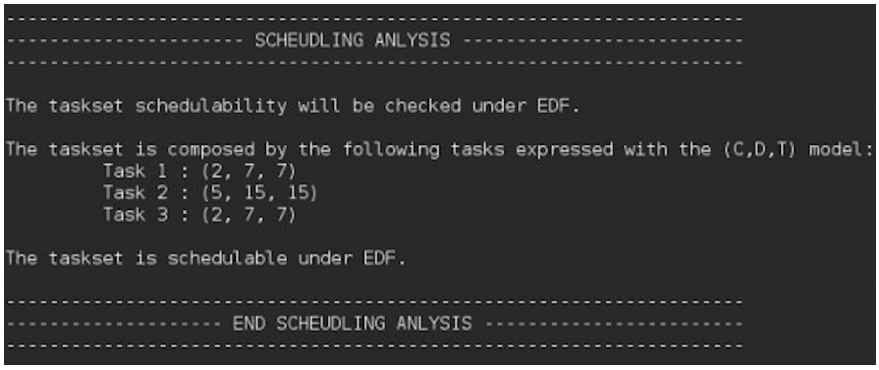
7.3.3 DM

For the completeness of information, we report the result under DM without performing once again all the computations:

```
------------------------------------------------------------------
---------------------- SCHEUDLING ANLYSIS ------------------------
------------------------------------------------------------------

The taskset schedulability will be checked under DM.

The taskset is composed by the following tasks expressed with the (C,D,T) model:
        Task 1 : (2, 7, 7)
        Task 3 : (2, 7, 7)
        Task 2 : (5, 15, 15)

The taskset is schedulable under DM.

------------------------------------------------------------------
------------------- END SCHEUDLING ANLYSIS -----------------------
------------------------------------------------------------------
```

7.3.4 EDF

Let's see what happens under a dynamic priority algorithm like EDF:



Let's verify the correctness of the tool apply the processor demand approach:

$$H = 15$$

$$L^* = \frac{(T_1 - D_1)U_1 + (T_2 - D_2)U_2 + (T_3 - D_3)U_3}{1 - U} = 0$$

$$D_{max} = 15$$
$$D = \{d_k \mid d_k \le min[H, max(D_{max}, L^*)]\} = \{d_k \mid d_k \le 15\} = \{7, 14, 15\}$$

$$dbf(7) = \lfloor \frac{7 + T_1 - D_1}{T_1} \rfloor C_1 + \lfloor \frac{7 + T_2 - D_2}{T_2} \rfloor C_2 + \lfloor \frac{7 + T_3 - D_3}{T_3} \rfloor C_3 = 2 + 2 = 4 \le 7$$

$$dbf(14) = \lfloor \frac{14 + T_1 - D_1}{T_1} \rfloor C_1 + \lfloor \frac{14 + T_2 - D_2}{T_2} \rfloor C_2 + \lfloor \frac{14 + T_3 - D_3}{T_3} \rfloor C_3 = 4 + 4 = 8 \le 14$$

$$dbf(15) = \lfloor \frac{15 + T_1 - D_1}{T_1} \rfloor C_1 + \lfloor \frac{15 + T_2 - D_2}{T_2} \rfloor C_2 + \lfloor \frac{15 + T_3 - D_3}{T_3} \rfloor C_3 = 4 + 5 + 4 = 13 \le 15$$

The processor demand functions is less than all the scheduling points and so the task set is schedulable.

7.3.5 Other results

Let's apply now the same tool to the task set 2 and let's see the results:

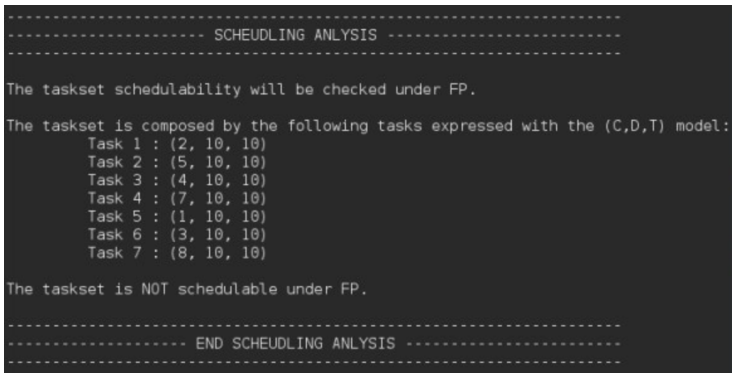
```
------------------------------------------------------------------
--------------------- SCHEUDLING ANLYSIS -------------------------
------------------------------------------------------------------

The taskset schedulability will be checked under FP.

The taskset is composed by the following tasks expressed with the (C,D,T) model:
        Task 1 : (2, 11, 11)
        Task 2 : (2, 16, 16)
        Task 3 : (3, 14, 14)
        Task 4 : (3, 11, 11)
        Task 5 : (2, 28, 28)

The taskset is schedulable under FP.

------------------------------------------------------------------
------------------- END SCHEUDLING ANLYSIS -----------------------
------------------------------------------------------------------
```

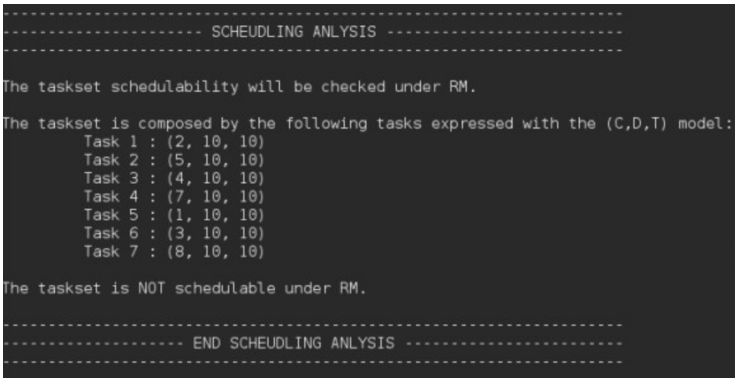
```
------------------------------------------------------------------
--------------------- SCHEUDLING ANLYSIS -------------------------
------------------------------------------------------------------

The taskset schedulability will be checked under RM.

The taskset is composed by the following tasks expressed with the (C,D,T) model:
        Task 1 : (2, 11, 11)
        Task 4 : (3, 11, 11)
        Task 3 : (3, 14, 14)
        Task 2 : (2, 16, 16)
        Task 5 : (2, 28, 28)

The taskset is schedulable under RM.

------------------------------------------------------------------
------------------- END SCHEUDLING ANLYSIS -----------------------
------------------------------------------------------------------
```

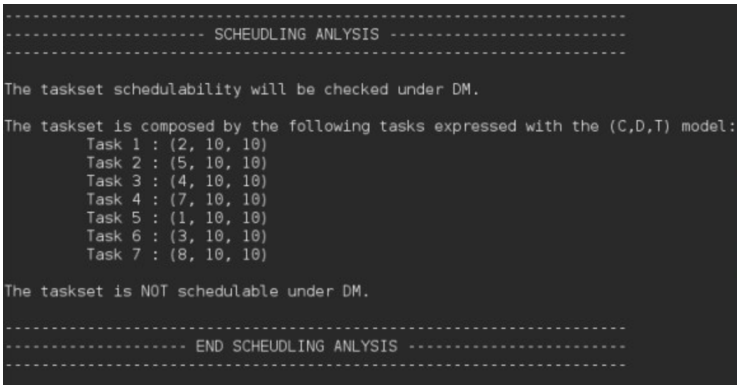
```
------------------------------------------------------------------
--------------------- SCHEUDLING ANLYSIS -------------------------
------------------------------------------------------------------

The taskset schedulability will be checked under DM.

The taskset is composed by the following tasks expressed with the (C,D,T) model:
        Task 1 : (2, 11, 11)
        Task 4 : (3, 11, 11)
        Task 3 : (3, 14, 14)
        Task 2 : (2, 16, 16)
        Task 5 : (2, 28, 28)

The taskset is schedulable under DM.

------------------------------------------------------------------
------------------- END SCHEUDLING ANLYSIS -----------------------
------------------------------------------------------------------
```

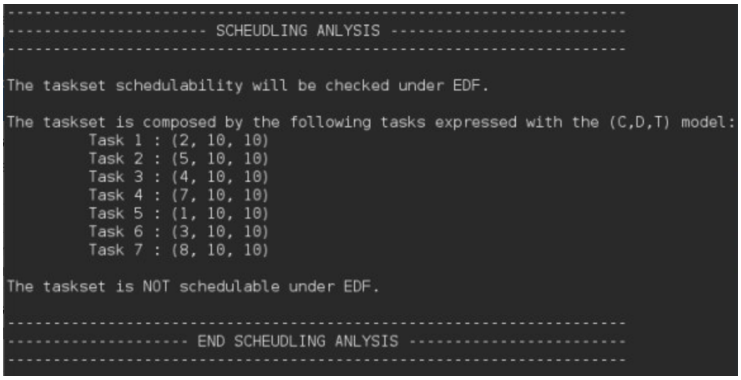
```
------------------------------------------------------------------
--------------------- SCHEUDLING ANLYSIS -------------------------
------------------------------------------------------------------

The taskset schedulability will be checked under EDF.

The taskset is composed by the following tasks expressed with the (C,D,T) model:
        Task 1 : (2, 11, 11)
        Task 2 : (2, 16, 16)
        Task 3 : (3, 14, 14)
        Task 4 : (3, 11, 11)
        Task 5 : (2, 28, 28)

The taskset is schedulable under EDF.

------------------------------------------------------------------
------------------- END SCHEUDLING ANLYSIS -----------------------
------------------------------------------------------------------
```

Let's apply now the same tool to the task set 3 and let's see the results:









In both the two cases all the computations are not reported because they are practically the same as already shown and the final result of the tool is correct.

7.4 Find periodic server

The third tool we are going to test is the *find_ps_analysis* tool that searches the periodic server able to schedule the taskset. Let's apply the tool to the task set 1. Once again we can choose to schedule according to 4 different scheduling algorithm, i.e. EDF, RM, DM and a fixed priority in the general sense.

7.4.1 Fixed priority

```
------------------------------------------------------------------
---------------- FIND PERIODIC SERVER ANALYSIS -------------------
------------------------------------------------------------------

The taskset schedulability will be checked under FP.

The taskset is composed by the following tasks expressed with the (C,D,T) model:
        Task 1 : (2, 7, 7)
        Task 2 : (5, 15, 15)
        Task 3 : (2, 7, 7)

Let's try to find a periodic server that can schedule the entire taskset under F
P. Sorry, it is impossible to find the desired periodic server.
------------------------------------------------------------------
-------------- END FIND PERIODIC SERVER ANALYSIS -----------------
------------------------------------------------------------------
```

Considering what stated in the chapter 4, we test all the periodic server with the period in in the following range:

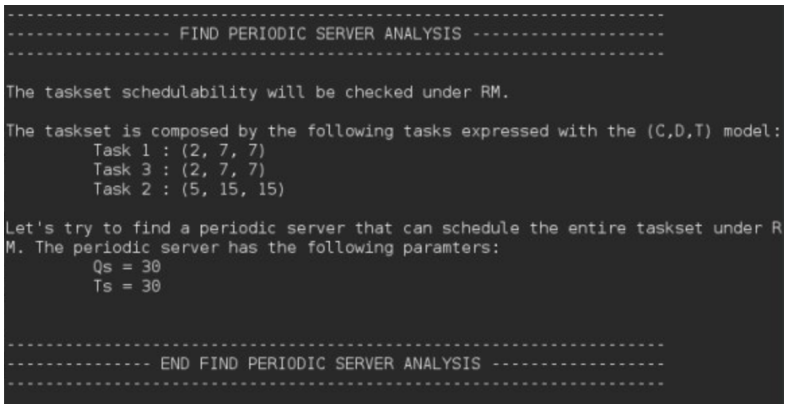
$$Ts \in [min\{T_i\}, 2 \cdot max\{T_i\}] \rightarrow Ts \in [7,30]$$

For each value of the period Ts we are testing, we check all the periodic server with budget in the range:

$$Qs \in [Ts \cdot U, Ts \cdot BW] \quad \text{where} \quad U = 0.9 \quad \text{and initially} \quad BW = 1$$

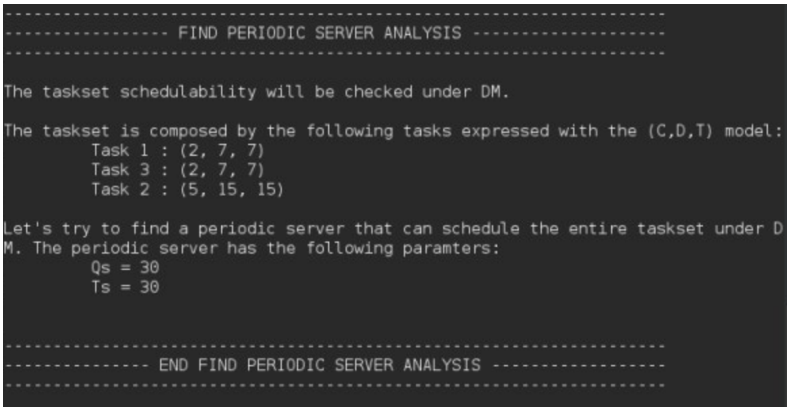
As we can see from the previous screenshot the tool was not able to find an useful periodic server within the ranges of budget and period explained in the previous chapters.

7.4.2 RM



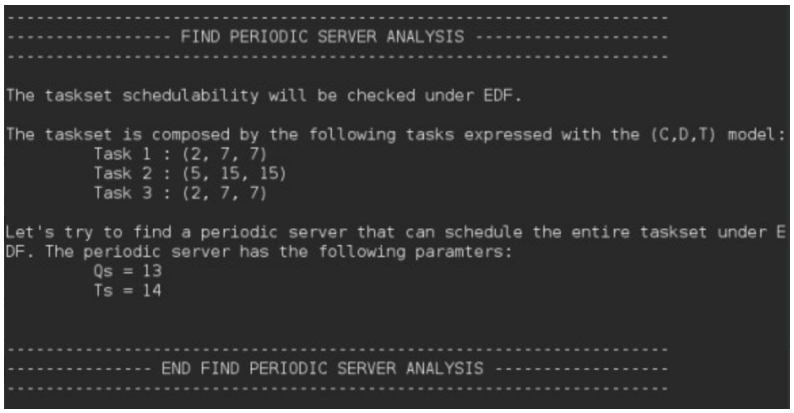
Also in this case the tool tests the same periodic server as before. However what changes is the scheduling algorithm and this means that under RM we are able to find a periodic server. Note that every time the tool finds a new server that has a bandwidth less than or equal to the current best server, we change the best one. The idea is to choose the server with greatest possible period because it will be the server with less impact in the system. In this case the tool tells us that in order to schedule the task set we need a server with bandwidth equal to 1. As already said, the tool chooses always the server with the greatest period being equal the bandwidth.

7.4.3 DM



The result is exactly the same of the RM case.

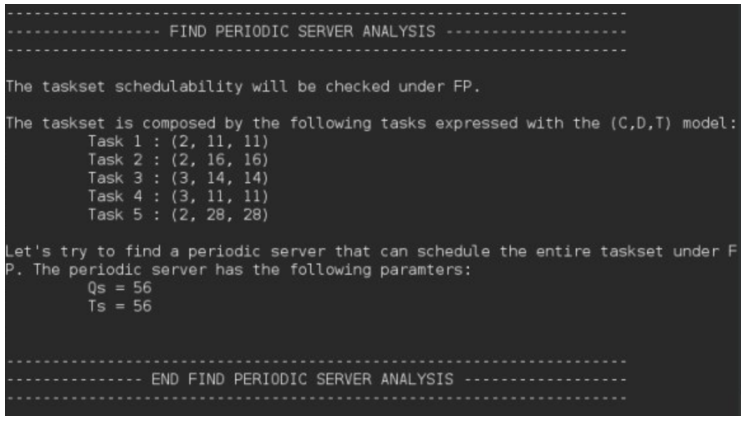
7.4.4 EDF

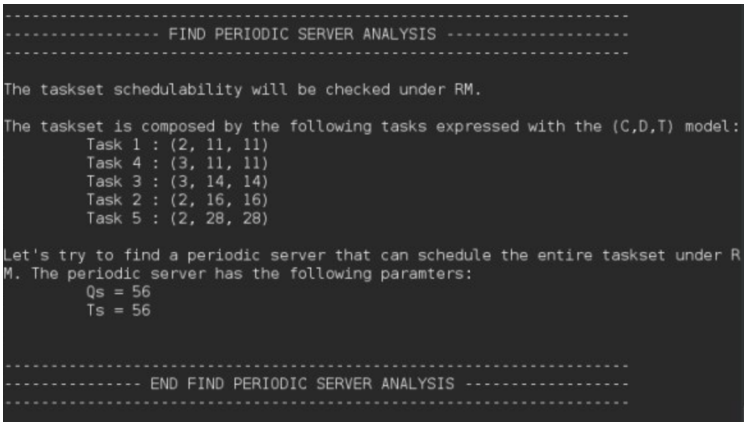


As we can see under EDF we are able to find a periodic server with bandwidth less than one (i.e. 0.93). This is due the fact that under EDF we are able to schedule the taskset in a more flexible way.

7.4.5 Other results

Let's apply now the same tool to the task set 2 and let's see the results:



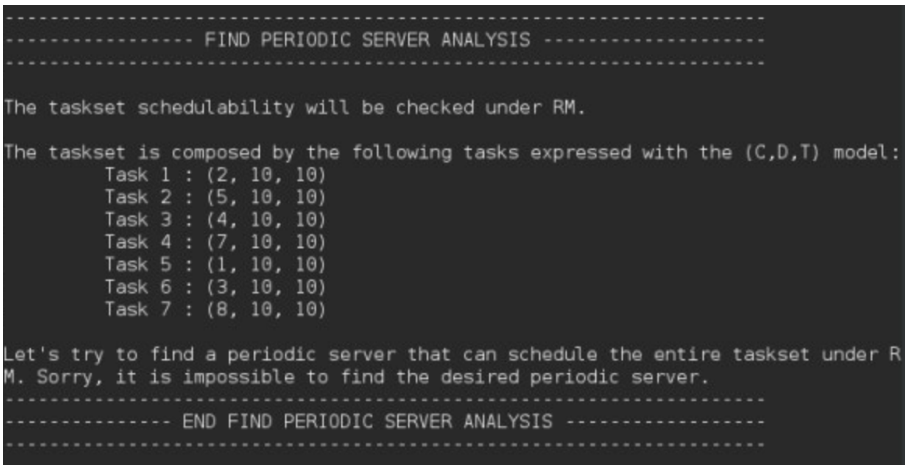


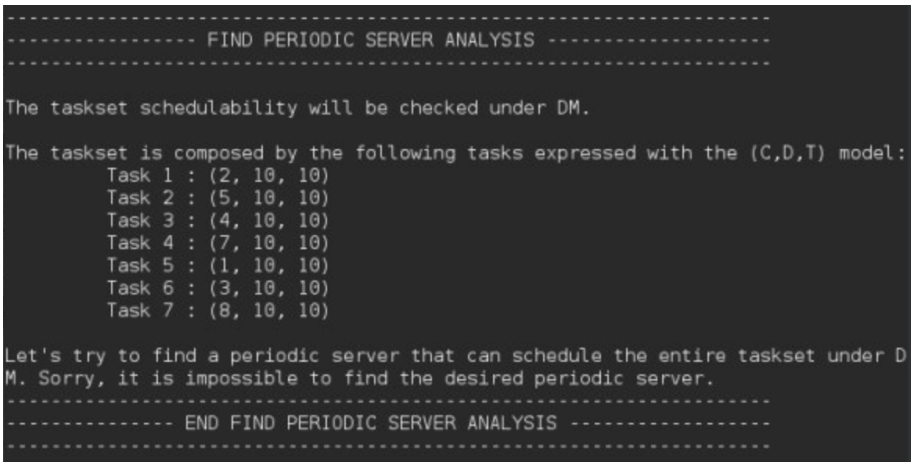
```
------------------------------------------------------------------
---------------- FIND PERIODIC SERVER ANALYSIS -------------------
------------------------------------------------------------------

The taskset schedulability will be checked under DM.

The taskset is composed by the following tasks expressed with the (C,D,T) model:
        Task 1 : (2, 11, 11)
        Task 4 : (3, 11, 11)
        Task 3 : (3, 14, 14)
        Task 2 : (2, 16, 16)
        Task 5 : (2, 28, 28)

Let's try to find a periodic server that can schedule the entire taskset under D
M. The periodic server has the following paramters:
        Qs = 56
        Ts = 56


------------------------------------------------------------------
-------------- END FIND PERIODIC SERVER ANALYSIS -----------------
------------------------------------------------------------------
```

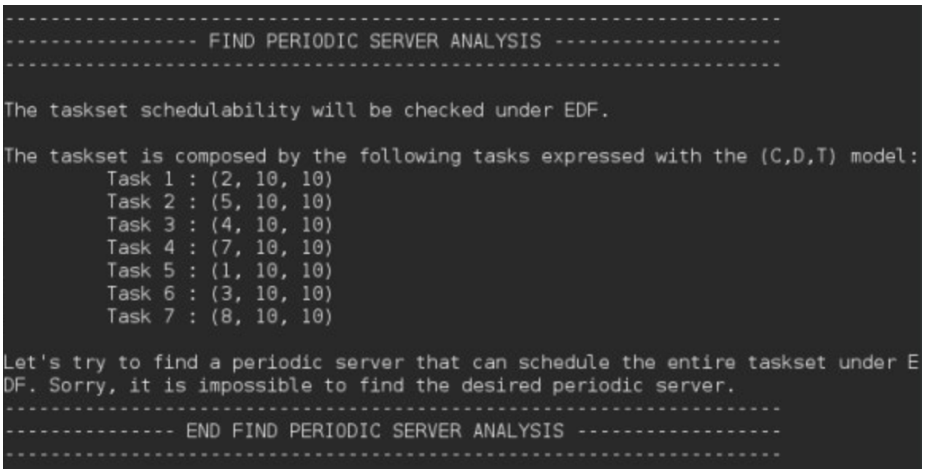
```
------------------------------------------------------------------
---------------- FIND PERIODIC SERVER ANALYSIS -------------------
------------------------------------------------------------------

The taskset schedulability will be checked under EDF.

The taskset is composed by the following tasks expressed with the (C,D,T) model:
        Task 1 : (2, 11, 11)
        Task 2 : (2, 16, 16)
        Task 3 : (3, 14, 14)
        Task 4 : (3, 11, 11)
        Task 5 : (2, 28, 28)

Let's try to find a periodic server that can schedule the entire taskset under E
DF. The periodic server has the following paramters:
        Qs = 13
        Ts = 15


------------------------------------------------------------------
-------------- END FIND PERIODIC SERVER ANALYSIS -----------------
------------------------------------------------------------------
```

Let's apply now the same tool to the task set 3 and let's see the results:

```
------------------------------------------------------------------
---------------- FIND PERIODIC SERVER ANALYSIS -------------------
------------------------------------------------------------------

The taskset schedulability will be checked under FP.

The taskset is composed by the following tasks expressed with the (C,D,T) model:
        Task 1 : (2, 10, 10)
        Task 2 : (5, 10, 10)
        Task 3 : (4, 10, 10)
        Task 4 : (7, 10, 10)
        Task 5 : (1, 10, 10)
        Task 6 : (3, 10, 10)
        Task 7 : (8, 10, 10)

Let's try to find a periodic server that can schedule the entire taskset under F
P. Sorry, it is impossible to find the desired periodic server.
------------------------------------------------------------------
-------------- END FIND PERIODIC SERVER ANALYSIS -----------------
------------------------------------------------------------------
```







In both the two cases all the reasoning are not reported because they are practically the same as already shown and the final result of the tool is correct.

7.5 Hierarchical analysis

The next tool we are going to test is the *h_analysis* tool that checks the schedulability of a periodic taskset with respect to budget and period of a periodic server. Let's apply the tool to the task set 1. Once again we can choose to schedule according to 4 different scheduling algorithm, i.e. EDF, RM, DM and a fixed priority in the general sense. In addition we choose to test the the tool using optimal budgets and periods computed by the *find_ps_analsys* tool. This means that for task set 1 we are going to use a periodic server with $Ts = 13$ and $Qs = 14$.

7.5.1 Fixed priority

Let's start by testing the schedulability under fixed priority choosing as priority order the one output by the random task set generator.

```
------------------------------------------------------------------
---------------- HIERARCHICAL SCHEDULING ANALYSIS ----------------
------------------------------------------------------------------

The taskset is composed by the following tasks expressed with the (C,D,T) model:
        Task 1 : (2, 7, 7)
        Task 2 : (5, 15, 15)
        Task 3 : (2, 7, 7)

The periodic server has the following paramters:
        Qs = 13
        Ts = 14


The taskset is NOT schedulable under FP with the specified periodic server (Qs =
 13, Ts = 14).
Let's try to find a periodic server that can schedule the entire taskset under F
P. Sorry, it is impossible to find the desired periodic server.
------------------------------------------------------------------
------------- END HIERARCHICAL SCHEDULING ANALYSIS ---------------
------------------------------------------------------------------
```

According to the tool the task set is not schedulable with the given periodic server and this must be true because the task set is just not schedulable in the general sense.

7.5.2 RM

```
------------------------------------------------------------------
---------------- HIERARCHICAL SCHEDULING ANALYSIS ----------------
------------------------------------------------------------------

The taskset is composed by the following tasks expressed with the (C,D,T) model:
        Task 1 : (2, 7, 7)
        Task 3 : (2, 7, 7)
        Task 2 : (5, 15, 15)

The periodic server has the following paramters:
        Qs = 13
        Ts = 14


The taskset is NOT schedulable under RM with the specified periodic server (Qs =
 13, Ts = 14).
Let's try to find a periodic server that can schedule the entire taskset under R
M. The periodic server has the following paramters:
        Qs = 30
        Ts = 30


------------------------------------------------------------------
------------- END HIERARCHICAL SCHEDULING ANALYSIS ---------------
------------------------------------------------------------------
```

In this case the task set was considered schedulable and so we could expect to find an useful periodic server. Remember that the tool *find_ps_analysis* told us that in order to schedule this task set we need a bandwidth equal to 1. This is not the case for our server and in fact this tool replies that the task set is not schedulable but then it is able to find a periodic server (the same found before).

7.5.3 DM

```
------------------------------------------------------------------
---------------- HIERARCHICAL SCHEDULING ANALYSIS -----------------
------------------------------------------------------------------

The taskset is composed by the following tasks expressed with the (C,D,T) model:
        Task 1 : (2, 7, 7)
        Task 3 : (2, 7, 7)
        Task 2 : (5, 15, 15)

The periodic server has the following paramters:
        Qs = 13
        Ts = 14


The taskset is NOT schedulable under DM with the specified periodic server (Qs =
 13, Ts = 14).
Let's try to find a periodic server that can schedule the entire taskset under D
M. The periodic server has the following paramters:
        Qs = 30
        Ts = 30


------------------------------------------------------------------
-------------- END HIERARCHICAL SCHEDULING ANALYSIS ---------------
------------------------------------------------------------------
```

What stated for RM is valid also in this case.

7.5.4 EDF

```
------------------------------------------------------------------
---------------- HIERARCHICAL SCHEDULING ANALYSIS -----------------
------------------------------------------------------------------

The taskset is composed by the following tasks expressed with the (C,D,T) model:
        Task 1 : (2, 7, 7)
        Task 2 : (5, 15, 15)
        Task 3 : (2, 7, 7)

The periodic server has the following paramters:
        Qs = 13
        Ts = 14


The taskset is schedulable under EDF with the specified periodic server (Qs = 13
, Ts = 14).

------------------------------------------------------------------
-------------- END HIERARCHICAL SCHEDULING ANALYSIS ---------------
------------------------------------------------------------------
```

We used the periodic server that was able to schedule the task set 1 under EDF according to the *find_ps_analysis* tool. As we can imagine, the *h_analysis* tool confirms our supposition.

Anyway, let's try to compute by hand the schedulability analysis:

$$H = 15$$

$$L^* = \frac{(T_1 - D_1)U_1 + (T_2 - D_2)U_2 + (T_3 - D_3)U_3}{1 - U} = 0$$

$$D_{max} = 15$$

$$D = \{d_k \mid d_k \leq min[H, max(D_{max}, L^*)]\} = \{d_k \mid d_k \leq 15\} = \{7, 14, 15\}$$
$$SBF = \{t \mid t = n \cdot Ts - Qs, n \geq 2, t < D_{max}\} = \{t \mid t = n \cdot 14 - 13, n \geq 2, t < 15\} = \emptyset$$

$$D \cup SBF = \{7, 14, 15\}$$

$$dbf(7) = \lfloor \frac{7 + T_1 - D_1}{T_1} \rfloor C_1 + \lfloor \frac{7 + T_2 - D_2}{T_2} \rfloor C_2 + \lfloor \frac{7 + T_3 - D_3}{T_3} \rfloor C_3 = 2 + 2 = 4 \leq sbf(7) = 5$$

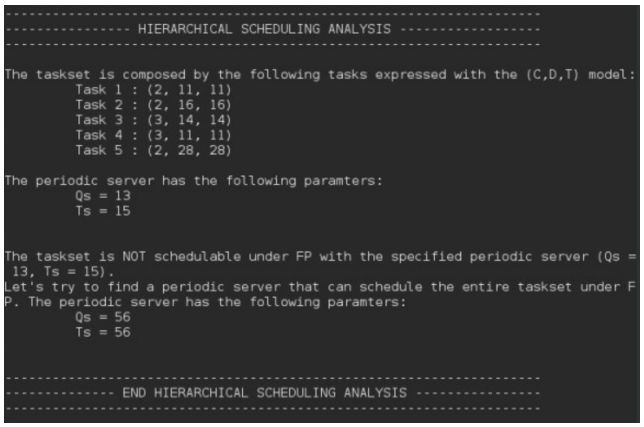
$$dbf(14) = \lfloor \frac{14 + T_1 - D_1}{T_1} \rfloor C_1 + \lfloor \frac{14 + T_2 - D_2}{T_2} \rfloor C_2 + \lfloor \frac{14 + T_3 - D_3}{T_3} \rfloor C_3 = 4 + 4 = 8 \leq sbf(14) = 12$$

$$dbf(15) = \lfloor \frac{15 + T_1 - D_1}{T_1} \rfloor C_1 + \lfloor \frac{15 + T_2 - D_2}{T_2} \rfloor C_2 + \lfloor \frac{15 + T_3 - D_3}{T_3} \rfloor C_3 = 4 + 5 + 4 = 13 \leq sbf(15) = 13$$

As we can see the demand bound function is less than or equal to the supplied bound function in all the testing points, so it follows that the task set is schedulable.

7.5.5 Other results

Let's apply now the same tool to the task set 2 with a periodic server with $Qs = 13$ and $Ts = 15$ and let's see the results:



```
------------------------------------------------------------------
---------------- HIERARCHICAL SCHEDULING ANALYSIS ------------------
------------------------------------------------------------------

The taskset is composed by the following tasks expressed with the (C,D,T) model:
        Task 1 : (2, 11, 11)
        Task 4 : (3, 11, 11)
        Task 3 : (3, 14, 14)
        Task 2 : (2, 16, 16)
        Task 5 : (2, 28, 28)

The periodic server has the following paramters:
        Qs = 13
        Ts = 15


The taskset is NOT schedulable under RM with the specified periodic server (Qs =
 13, Ts = 15).
Let's try to find a periodic server that can schedule the entire taskset under R
M. The periodic server has the following paramters:
        Qs = 56
        Ts = 56


------------------------------------------------------------------
------------- END HIERARCHICAL SCHEDULING ANALYSIS ---------------
------------------------------------------------------------------
```

```
------------------------------------------------------------------
---------------- HIERARCHICAL SCHEDULING ANALYSIS ------------------
------------------------------------------------------------------

The taskset is composed by the following tasks expressed with the (C,D,T) model:
        Task 1 : (2, 11, 11)
        Task 4 : (3, 11, 11)
        Task 3 : (3, 14, 14)
        Task 2 : (2, 16, 16)
        Task 5 : (2, 28, 28)

The periodic server has the following paramters:
        Qs = 13
        Ts = 15


The taskset is NOT schedulable under DM with the specified periodic server (Qs =
 13, Ts = 15).
Let's try to find a periodic server that can schedule the entire taskset under D
M. The periodic server has the following paramters:
        Qs = 56
        Ts = 56


------------------------------------------------------------------
------------- END HIERARCHICAL SCHEDULING ANALYSIS ---------------
------------------------------------------------------------------
```

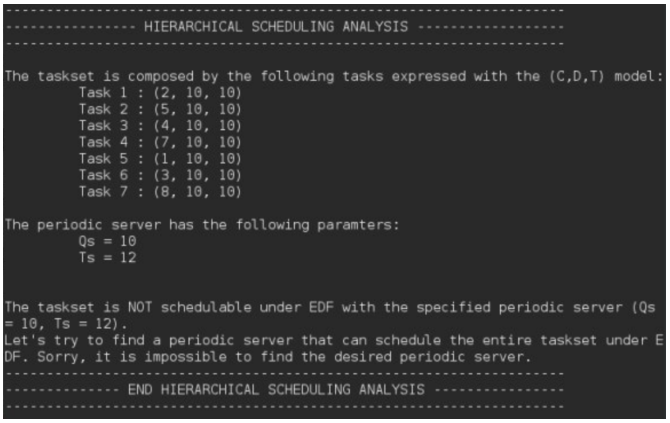
```
------------------------------------------------------------------
---------------- HIERARCHICAL SCHEDULING ANALYSIS ------------------
------------------------------------------------------------------

The taskset is composed by the following tasks expressed with the (C,D,T) model:
        Task 1 : (2, 11, 11)
        Task 2 : (2, 16, 16)
        Task 3 : (3, 14, 14)
        Task 4 : (3, 11, 11)
        Task 5 : (2, 28, 28)

The periodic server has the following paramters:
        Qs = 13
        Ts = 15


The taskset is schedulable under EDF with the specified periodic server (Qs = 13
, Ts = 15).

------------------------------------------------------------------
------------- END HIERARCHICAL SCHEDULING ANALYSIS ---------------
------------------------------------------------------------------
```

Let's apply now the same tool to the task set 3 with a periodic server with $Qs = 10$ and $Ts = 12$ and let's see the results:

```
------------------------------------------------------------------
---------------- HIERARCHICAL SCHEDULING ANALYSIS ----------------
------------------------------------------------------------------

The taskset is composed by the following tasks expressed with the (C,D,T) model:
        Task 1 : (2, 10, 10)
        Task 2 : (5, 10, 10)
        Task 3 : (4, 10, 10)
        Task 4 : (7, 10, 10)
        Task 5 : (1, 10, 10)
        Task 6 : (3, 10, 10)
        Task 7 : (8, 10, 10)

The periodic server has the following paramters:
        Qs = 10
        Ts = 12


The taskset is NOT schedulable under FP with the specified periodic server (Qs =
 10, Ts = 12).
Let's try to find a periodic server that can schedule the entire taskset under F
P. Sorry, it is impossible to find the desired periodic server.
------------------------------------------------------------------
-------------- END HIERARCHICAL SCHEDULING ANALYSIS --------------
------------------------------------------------------------------
```

```
------------------------------------------------------------------
---------------- HIERARCHICAL SCHEDULING ANALYSIS ----------------
------------------------------------------------------------------

The taskset is composed by the following tasks expressed with the (C,D,T) model:
        Task 1 : (2, 10, 10)
        Task 2 : (5, 10, 10)
        Task 3 : (4, 10, 10)
        Task 4 : (7, 10, 10)
        Task 5 : (1, 10, 10)
        Task 6 : (3, 10, 10)
        Task 7 : (8, 10, 10)

The periodic server has the following paramters:
        Qs = 10
        Ts = 12


The taskset is NOT schedulable under RM with the specified periodic server (Qs =
 10, Ts = 12).
Let's try to find a periodic server that can schedule the entire taskset under R
M. Sorry, it is impossible to find the desired periodic server.
------------------------------------------------------------------
-------------- END HIERARCHICAL SCHEDULING ANALYSIS --------------
------------------------------------------------------------------
```

```
------------------------------------------------------------------
---------------- HIERARCHICAL SCHEDULING ANALYSIS ----------------
------------------------------------------------------------------

The taskset is composed by the following tasks expressed with the (C,D,T) model:
        Task 1 : (2, 10, 10)
        Task 2 : (5, 10, 10)
        Task 3 : (4, 10, 10)
        Task 4 : (7, 10, 10)
        Task 5 : (1, 10, 10)
        Task 6 : (3, 10, 10)
        Task 7 : (8, 10, 10)

The periodic server has the following paramters:
        Qs = 10
        Ts = 12


The taskset is NOT schedulable under DM with the specified periodic server (Qs =
 10, Ts = 12).
Let's try to find a periodic server that can schedule the entire taskset under D
M. Sorry, it is impossible to find the desired periodic server.
------------------------------------------------------------------
-------------- END HIERARCHICAL SCHEDULING ANALYSIS --------------
------------------------------------------------------------------
```



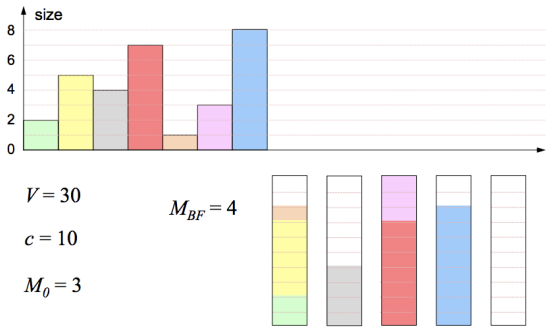
In both the two cases all the reasoning are not reported because they are practically the same as already shown and the final result of the tool is correct.

7.6 M-CPU analysis

The last tool we are going to test is the *mcpu_analysis* tool that checks the schedulability of a periodic taskset running on a multiprocessor machine. Let's apply the tool to the task set 3 being the one described in [8]. Once again we can choose to schedule according to 4 different scheduling algorithm, i.e. EDF, RM, DM and a fixed priority in the general sense. In addition we can choose between 4 different heuristic allocation algorithms, i.e. BF, FF, NF, WF and FFD. Let's see first all the allocation algorithms under EDF (as in [8]) and then we move to see the results under RM. Note that for RM is not possible to use the FFD algorithm and that in BF and WF algorithms we need to use the modified utilization based on the workload function. In both the cases we are going to test the tool with the virtual machine 1 and then we will some results using virtual machine 2.

7.6.1 BF under EDF

Let's see the results as presented in [8]:



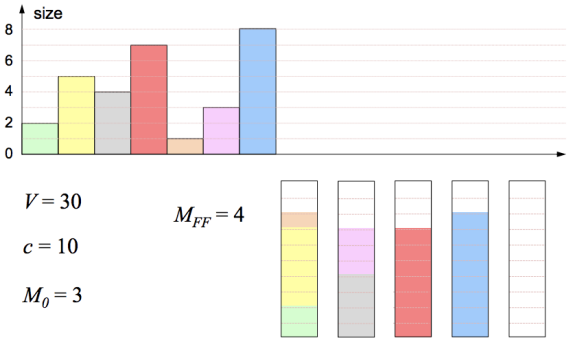
The result of the tool is instead the following one:

```
The taskset is schedulable under multi-CPU partitioned scheduling with the specified cpus, using BEST FIT as allocation algorithm and using EDF as scheduling algorithm.
The computational load on the vm is the following:
        Cpu 1 :
                U : 0.800000
                Task 1 : (2, 10, 10)
                Task 2 : (5, 10, 10)
                Task 5 : (1, 10, 10)
        Cpu 2 :
                U : 0.400000
                Task 3 : (4, 10, 10)
        Cpu 3 :
                U : 1.000000
                Task 4 : (7, 10, 10)
                Task 6 : (3, 10, 10)
        Cpu 4 :
                U : 0.800000
                Task 7 : (8, 10, 10)
        Cpu 5 :
                U : 0.000000

        Cpu 1 : [11222225]
        Cpu 2 : [3333]
        Cpu 3 : [4444444666]
        Cpu 4 : [77777777]
        Cpu 5 : []
```

As we can see from the last print of the tool, the final allocation of tasks to the different cpus is exactly the desired one.

7.6.2 FF under EDF

Let's see the results as presented in [8]:



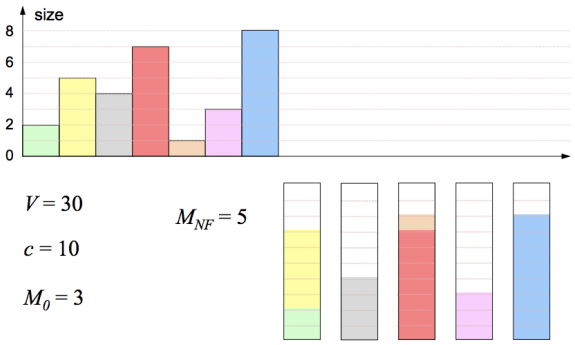
The result of the tool is instead the following one:

```
The taskset is schedulable under multi-CPU partitioned scheduling with the specified cpus, using FIRST FIT as allocation algorithm and using EDF as scheduling algorithm.
The computational load on the vm is the following:
        Cpu 1 :
                U : 0.800000
                Task 1 : (2, 10, 10)
                Task 2 : (5, 10, 10)
                Task 5 : (1, 10, 10)
        Cpu 2 :
                U : 0.700000
                Task 3 : (4, 10, 10)
                Task 6 : (3, 10, 10)
        Cpu 3 :
                U : 0.700000
                Task 4 : (7, 10, 10)
        Cpu 4 :
                U : 0.800000
                Task 7 : (8, 10, 10)
        Cpu 5 :
                U : 0.000000

        Cpu 1 : [11222225]
        Cpu 2 : [3333666]
        Cpu 3 : [4444444]
        Cpu 4 : [77777777]
        Cpu 5 : []
```

The result is correct also in this case.

7.6.3 NF under EDF

Let's see the results as presented in [8]:



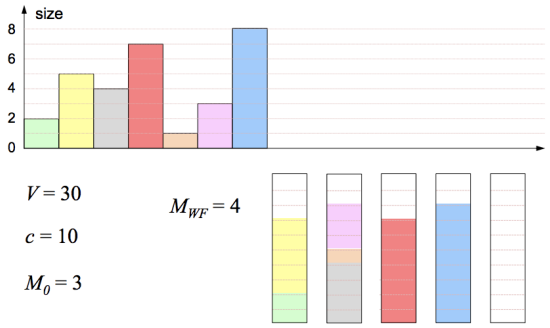
The result of the tool is instead the following one:

```
The taskset is schedulable under multi-CPU partitioned scheduling with the specified cpus, using NEXT FIT as allocation algorithm and using EDF as scheduling algorithm.
The computational load on the vm is the following:
        Cpu 1 :
                U : 0.700000
                Task 1 : (2, 10, 10)
                Task 2 : (5, 10, 10)
        Cpu 2 :
                U : 0.400000
                Task 3 : (4, 10, 10)
        Cpu 3 :
                U : 0.800000
                Task 4 : (7, 10, 10)
                Task 5 : (1, 10, 10)
        Cpu 4 :
                U : 0.300000
                Task 6 : (3, 10, 10)
        Cpu 5 :
                U : 0.800000
                Task 7 : (8, 10, 10)

        Cpu 1 : [1122222]
        Cpu 2 : [3333]
        Cpu 3 : [44444445]
        Cpu 4 : [666]
        Cpu 5 : [77777777]
```

The result is correct also in this case.

7.6.4 WF under EDF

In this case the result presented in [8] refers to a different implementation of the worst fit algorithm in which we assign the task to the used CPU with the largest empty space and we don't start a new CPU if we can schedule it in a previously used CPU.



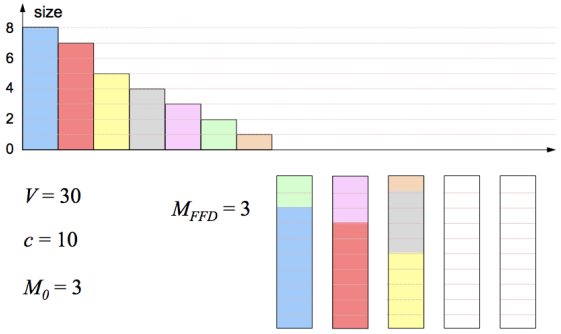
The result of the tool is instead the following one:

```
The taskset is schedulable under multi-CPU partitioned scheduling with the specified cpus, using WORST FIT as allocation algorithm and using EDF as scheduling algorithm.
The computational load on the vm is the following:
        Cpu 1 :
                U : 1.000000
                Task 1 : (2, 10, 10)
                Task 7 : (8, 10, 10)
        Cpu 2 :
                U : 0.500000
                Task 2 : (5, 10, 10)
        Cpu 3 :
                U : 0.400000
                Task 3 : (4, 10, 10)
        Cpu 4 :
                U : 0.700000
                Task 4 : (7, 10, 10)
        Cpu 5 :
                U : 0.400000
                Task 5 : (1, 10, 10)
                Task 6 : (3, 10, 10)

        Cpu 1 : [1177777777]
        Cpu 2 : [22222]
        Cpu 3 : [3333]
        Cpu 4 : [4444444]
        Cpu 5 : [5666]
```

Even if the result is different from what stated in [8], the final result is correct.

7.6.5 FFD under EDF

Let's see the results as presented in [8]:

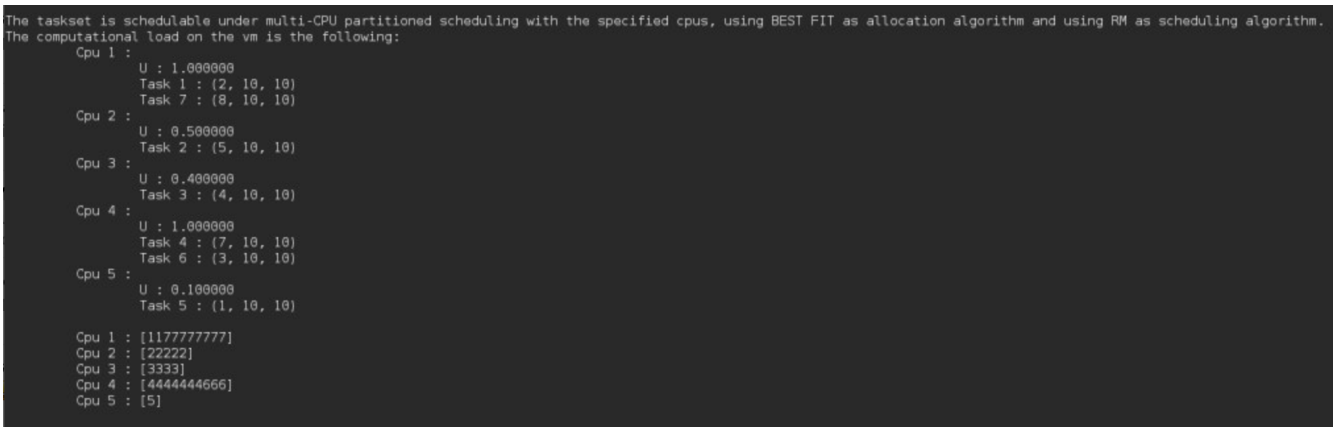


The result of the tool is instead the following one:

```
The taskset is schedulable under multi-CPU partitioned scheduling with the specified cpus, using FIRST FIT DECREASING as allocation algorithm and using EDF as scheduling algorithm.
The computational load on the vm is the following:
        Cpu 1 :
                U : 1.000000
                Task 7 : (8, 10, 10)
                Task 1 : (2, 10, 10)
        Cpu 2 :
                U : 1.000000
                Task 4 : (7, 10, 10)
                Task 6 : (3, 10, 10)
        Cpu 3 :
                U : 1.000000
                Task 2 : (5, 10, 10)
                Task 3 : (4, 10, 10)
                Task 5 : (1, 10, 10)
        Cpu 4 :
                U : 0.000000
        Cpu 5 :
                U : 0.000000

        Cpu 1 : [7777777711]
        Cpu 2 : [4444444666]
        Cpu 3 : [2222233335]
        Cpu 4 : []
        Cpu 5 : []
```

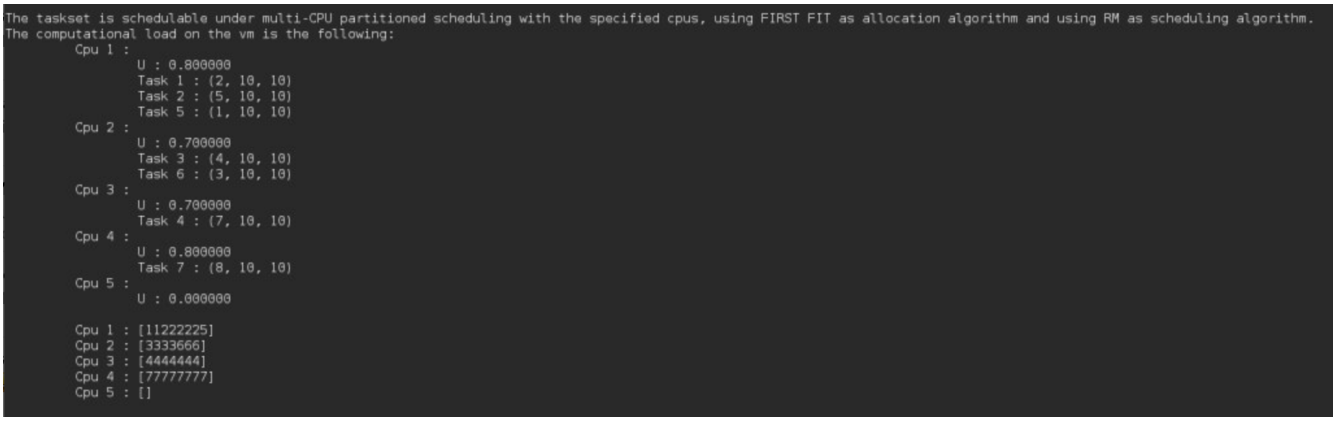
The result is correct also in this case.

7.6.6 BF under RM



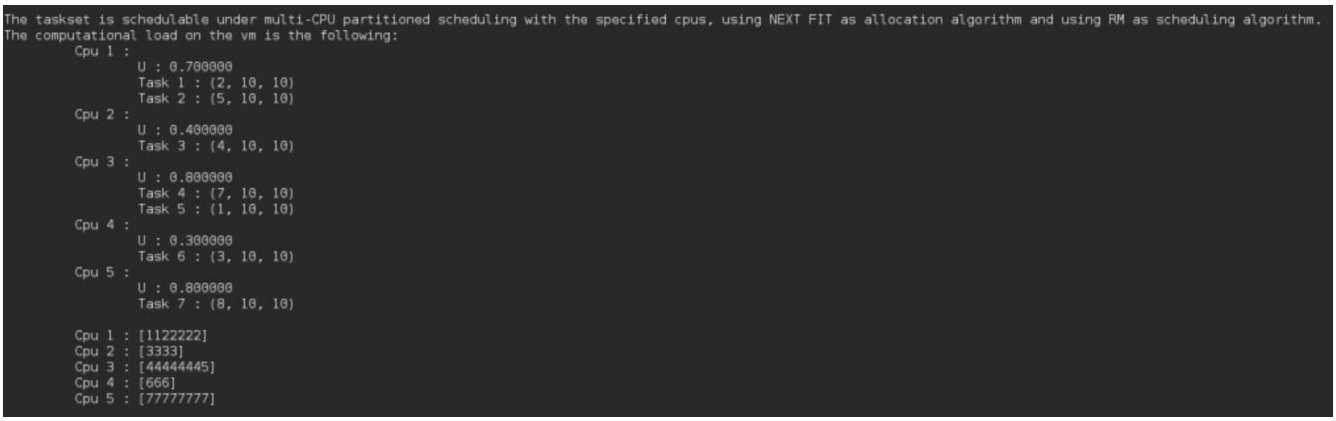
In this case the final result is different from the one at 7.6.1 because in case of rate monotonic, or in general of a fixed priority algorithm, we need to use the modified utilization of the task set and this produces a different allocation.

7.6.7 FF under RM



As we can imagine, in this case we are going to use the same tools used for allocate tasks under EDF and so the final result is exactly the same.

7.6.8 NF under RM



As for FF, the result is exactly the same of EDF.

7.6.9 WF under RM

Once again we use the modified utilization and so we have a different final allocation.

7.6.10 Other results

Let's see what happens if we try to schedule the task set 3 under EDF on the virtual machine 2 using BF as allocation algorithm:

```
The taskset is schedulable under multi-CPU partitioned scheduling with the specified cpus, using BEST FIT as allocation algorithm and using EDF as scheduling algorithm.
The computational load on the vm is the following:
        Cpu 1 :
                U : 0.800000
                Task 1 : (2, 10, 10)
                Task 2 : (5, 10, 10)
                Task 5 : (1, 10, 10)
        Cpu 2 :
                U : 0.700000
                Task 3 : (4, 10, 10)
                Task 6 : (3, 10, 10)
        Cpu 3 :
                U : 0.700000
                Task 4 : (7, 10, 10)
        Cpu 4 :
                U : 0.800000
                Task 7 : (8, 10, 10)
        Cpu 5 :
                U : 0.000000

        Cpu 1 : [11222225]
        Cpu 2 : [3333666]
        Cpu 3 : [4444444]
        Cpu 4 : [77777777]
        Cpu 5 : []

Cpu 1 : finding a periodic server for the taskset. The periodic server has the following paramters:
        Qs = 9
        Ts = 10

Cpu 2 : finding a periodic server for the taskset. The periodic server has the following paramters:
        Qs = 9
        Ts = 10

Cpu 3 : finding a periodic server for the taskset. The periodic server has the following paramters:
        Qs = 9
        Ts = 10

Cpu 4 : finding a periodic server for the taskset. The periodic server has the following paramters:
        Qs = 9
        Ts = 10
```

As we can see from the result, the tool tries to find the best allocation using the simple scheduling analysis (chapter 2). Once the tool finds the best allocation according to BF (the same we obtained on virtual machine 1), it tries to find the best periodic server able to schedule the task set for each cpu.

Let's try to schedule the task set 1 under EDF on the virtual machine 1 using BF as allocation algorithm:

```
The taskset is schedulable under multi-CPU partitioned scheduling with the specified cpus, using BEST FIT as allocation algorithm and using EDF as scheduling algorithm.
The computational load on the vm is the following:
        Cpu 1 :
                U : 0.904762
                Task 1 : (2, 7, 7)
                Task 2 : (5, 15, 15)
                Task 3 : (2, 7, 7)
        Cpu 2 :
                U : 0.000000
        Cpu 3 :
                U : 0.000000
        Cpu 4 :
                U : 0.000000
        Cpu 5 :
                U : 0.000000

        Cpu 1 : [112222233]
        Cpu 2 : []
        Cpu 3 : []
        Cpu 4 : []
        Cpu 5 : []
```

Let's try to schedule the task set 2 under EDF on the virtual machine 1 using BF as allocation algorithm:

```
The taskset is schedulable under multi-CPU partitioned scheduling with the specified cpus, using BEST FIT as allocation algorithm and using EDF as scheduling algorithm.
The computational load on the vm is the following:
        Cpu 1 :
                U : 0.865260
                Task 1 : (2, 11, 11)
                Task 2 : (2, 16, 16)
                Task 3 : (3, 14, 14)
                Task 4 : (3, 11, 11)
                Task 5 : (2, 28, 28)
        Cpu 2 :
                U : 0.000000
        Cpu 3 :
                U : 0.000000
        Cpu 4 :
                U : 0.000000
        Cpu 5 :
                U : 0.000000

        Cpu 1 : [112233344455]
        Cpu 2 : []
        Cpu 3 : []
        Cpu 4 : []
        Cpu 5 : []
```

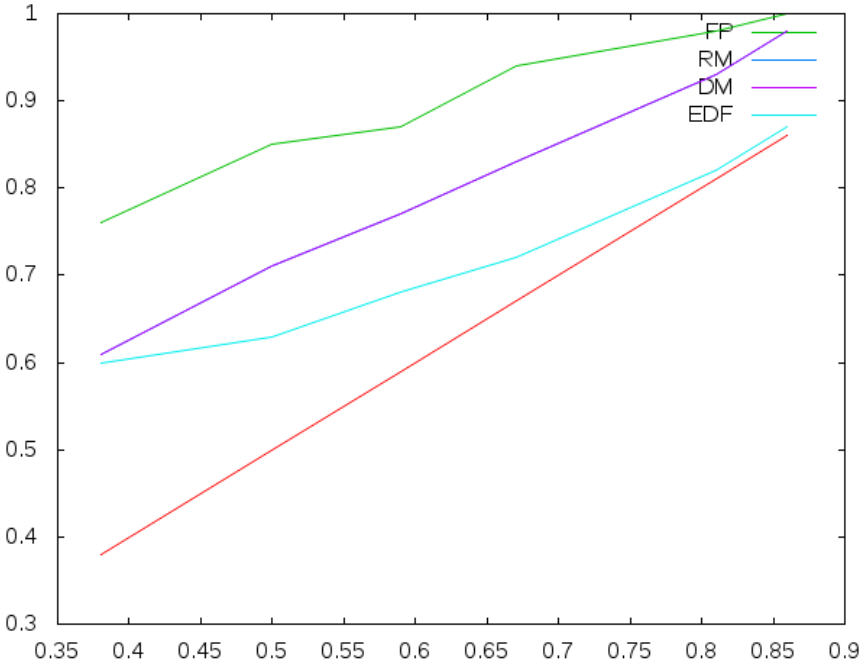
In both the last two cases the results are quite similar to the others and the final result of the tool is correct.

7.7 Comparison

One of the most useful feature we can exploit from the tool is the fact that we can use it in order to compare the schedulability of task sets under different algorithms. In particular the idea is to create a script able to plot some parameters useful to find the differences between different algorithms.

7.7.1 Scheduling algorithms

In this case the objective is to compare the server bandwidth needed in order to schedule the task set changing the scheduling algorithms and the utilization of the taskset. In particular the idea is to vary the utilization of the task set from 0.4 to 0.9 with step 0.1 and for each value we create 10 task sets. Then, for each task set, we find the periodic server using the *find_ps_analysis* and we compute the related bandwidth. Once computed the server bandwidths for the 10 task sets, we compute the mean of the values. At the end of this stage, we have an average server bandwidth for all the utilizations. Finally, the idea is to repeat this reasoning for all the possible scheduling algorithms. At the end, we can try to plot on the x axis the different utilizations (from 0.4 to 0.9) and on the y axis the different bandwidth (from 0 to 1). Of course, we will find 4 functions, one for each scheduling algorithm. The result of the described testing toll is the following one:

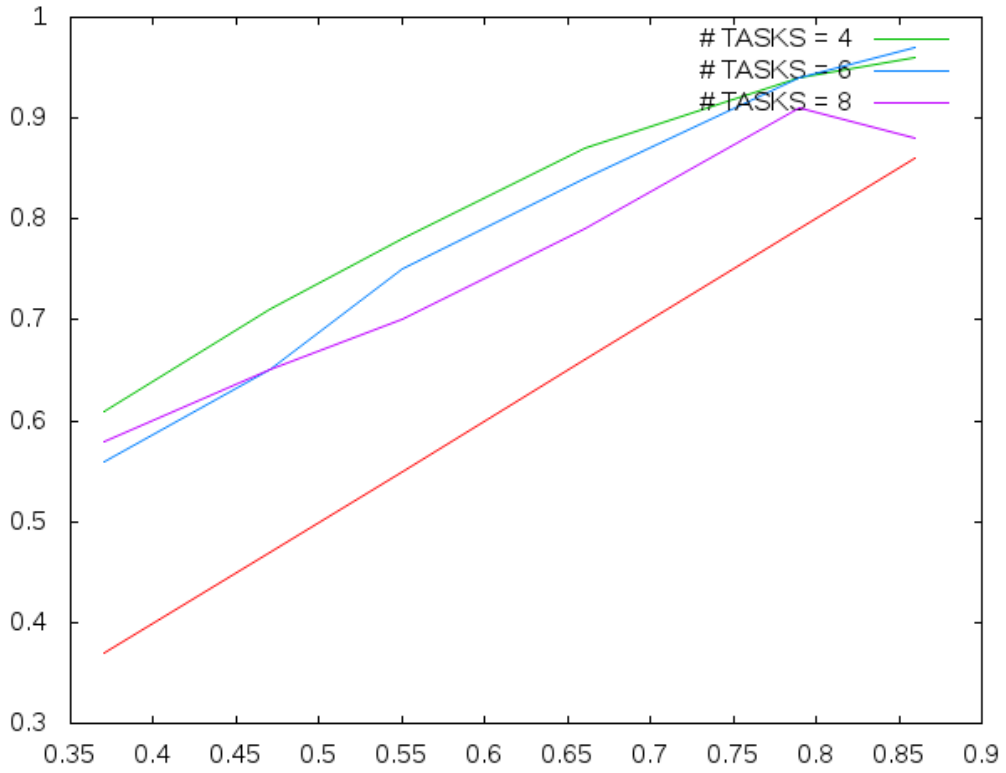


As we can expect, the bandwidth needed increases proportionally with the utilization of the task set. Moreover we can see that EDF, requires in general less bandwidths than fixed priority algorithms. Note that RM and DM have the same results because the task sets automatically generated have always periods equal to deadlines (i.e. no difference in ordering under RM or DM). The results under FP depends of course on the random order of the tasks when they are generated. The red line is the limit under which we cannot go because it represents the periodic server with bandwidth equal to the utilization of the task set.

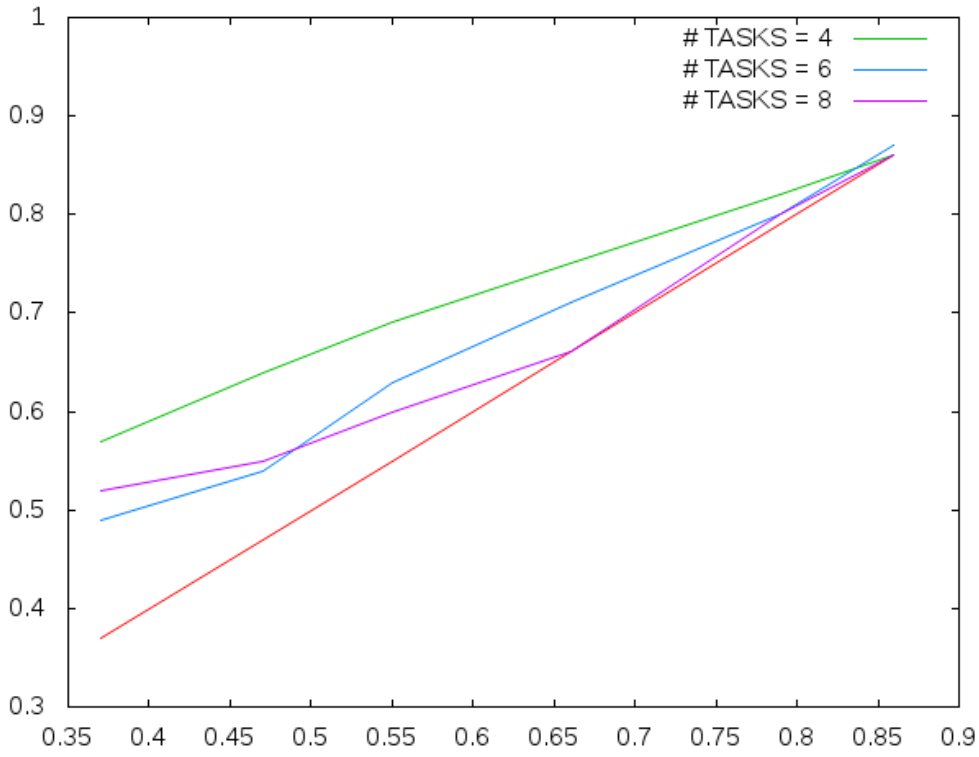
7.7.2 Task number

In this case the objective is to compare the server bandwidth needed to schedule the task set changing the scheduling algorithms, the utilization of the taskset and the dimension of the task set. In particular the basic idea is the same of the previous test. The difference is that in this case we produce a different plot for each scheduling algorithm. In each plot we can compare the server bandwidths for different dimensions of the taskset. Of course we are going to report only the cases RM and EDF because FP is in general an useless information (because it depends on a random order) and DM is the same of RM.

RM:



EDF:

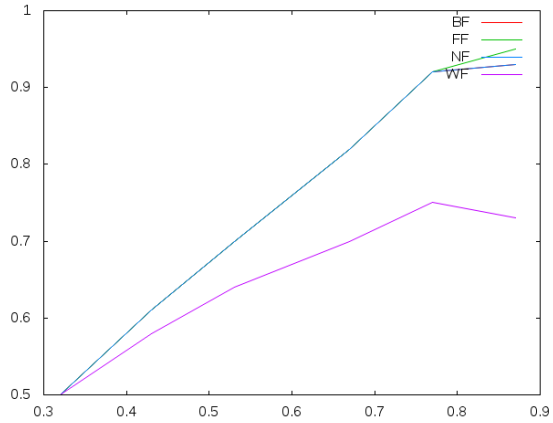


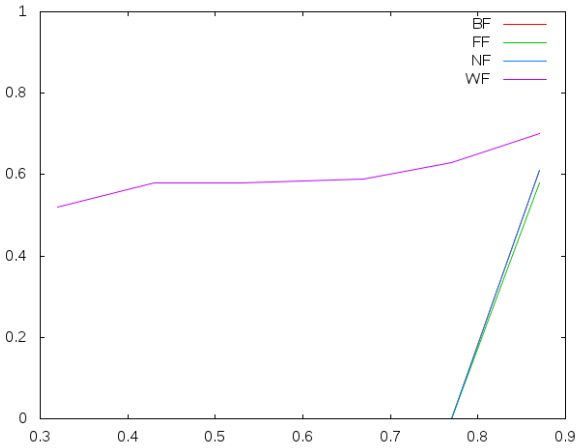
In both the cases we can see that increasing the number of tasks in the task set the server bandwidth needed tends to decrease. This is probably due to the fact that we increase the number of tasks keeping fixed the utilization. This means that we obtain lots of small tasks (some tasks have even computational time equal to 0) that are easier to schedule.

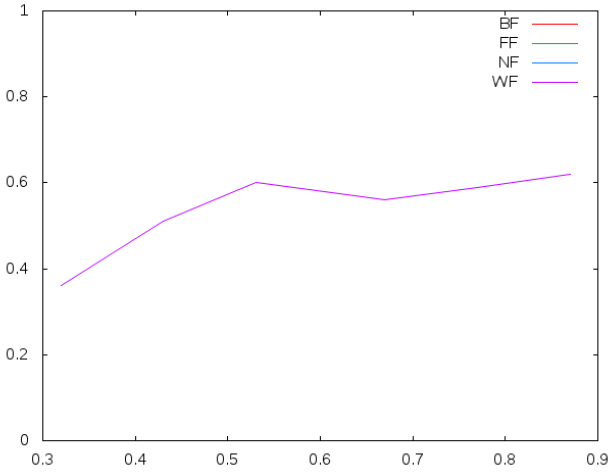
7.7.3 Multiple CPU

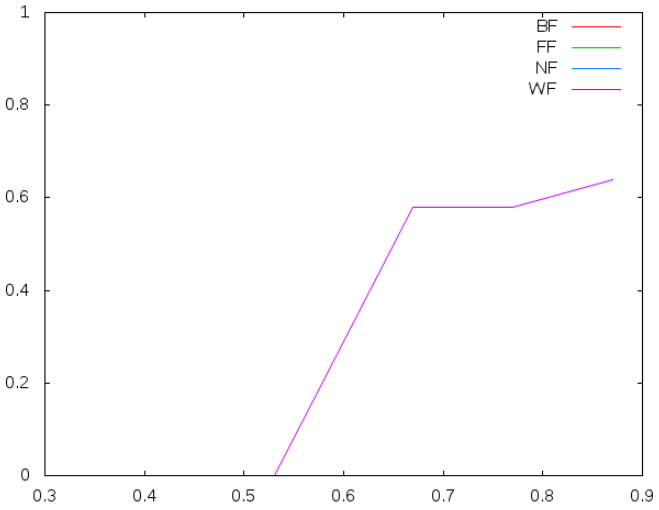
In this case the objective is to compare the server bandwidth needed to schedule the task set changing the scheduling algorithms, the allocation algorithm and the utilization of the taskset. In particular the basic idea is always the same but the difference is that in this case we produce a different plot for each scheduling algorithm and for each CPU. In each plot we can compare the server bandwidth needed on the specific CPU for different allocation algorithms. As in the previous case we are going to report the results only for RM and EDF. Note that we are considering the virtual machine 2 that has 5 CPUs. This means that for each scheduling algorithm, we will see 5 plots, one for each CPU.

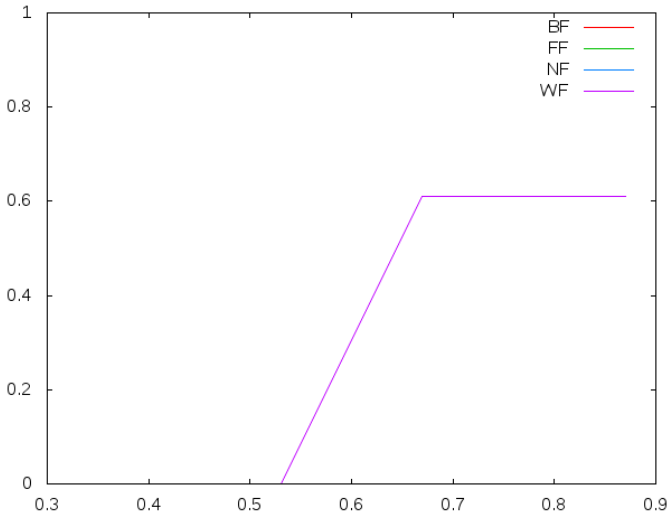
RM:


BF
FF
NF
WF
1
0.9
0.8
0.7
0.6
0.5
0.3
0.4
0.5
0.6
0.7
0.8
0.9

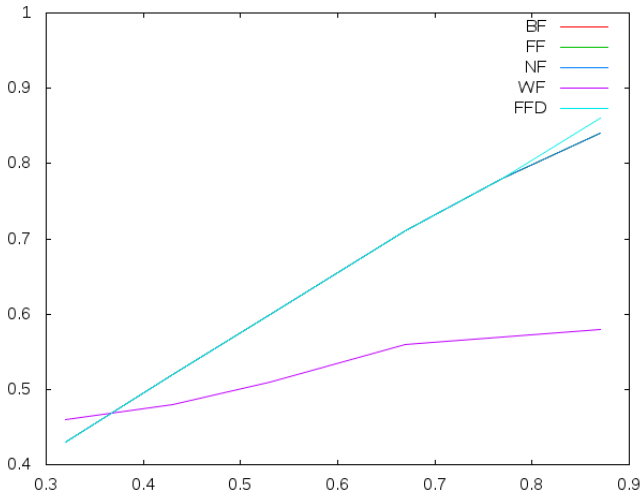

BF
FF
NF
WF
1
0.8
0.6
0.4
0.2
0
0.3
0.4
0.5
0.6
0.7
0.8
0.9

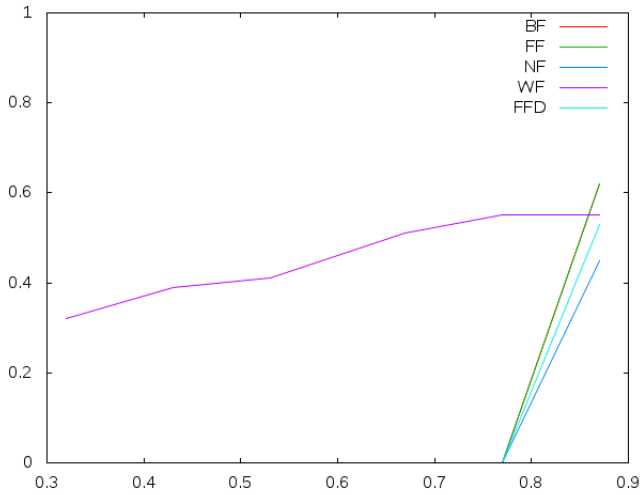

BF
FF
NF
WF
1
0.8
0.6
0.4
0.2
0
0.3
0.4
0.5
0.6
0.7
0.8
0.9

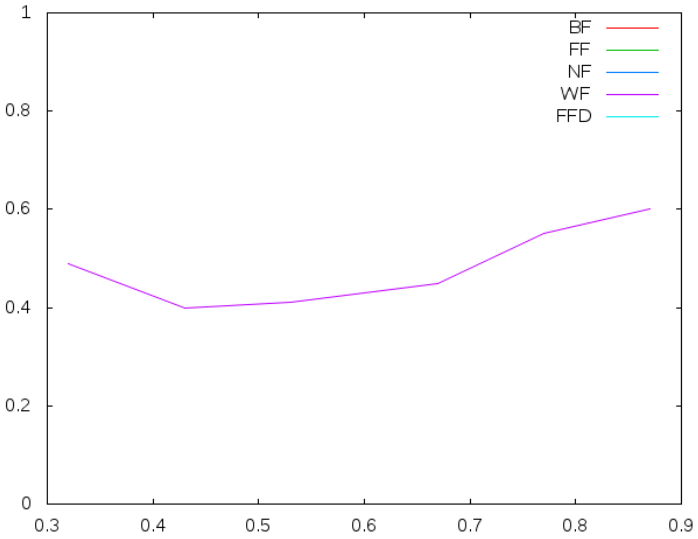

BF
FF
NF
WF
1
0.8
0.6
0.4
0.2
0
0.3
0.4
0.5
0.6
0.7
0.8
0.9

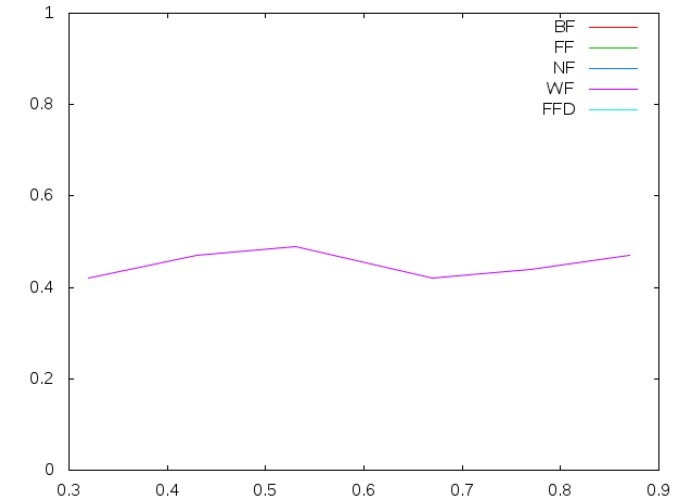

BF
FF
NF
WF
1
0.8
0.6
0.4
0.2
0
0.3
0.4
0.5
0.6
0.7
0.8
0.9

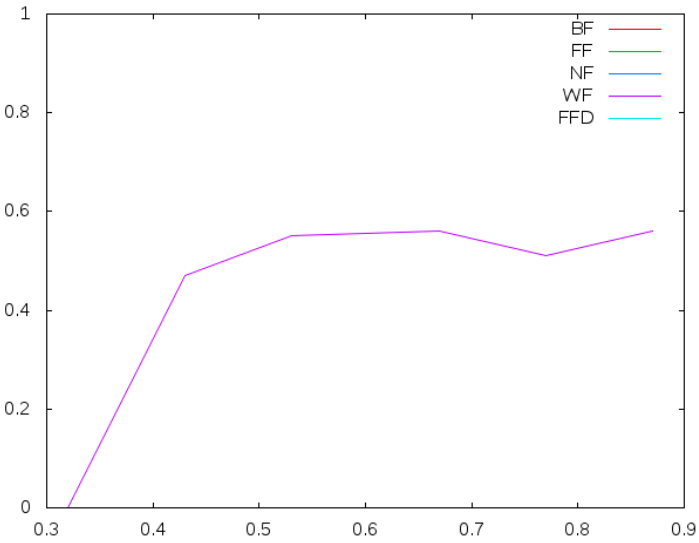
EDF:


BF
FF
NF
WF
FFD
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.4
0.5
0.6
0.7
0.8
0.9


BF
FF
NF
WF
FFD
1
0.8
0.6
0.4
0.2
0
0.3
0.4
0.5
0.6
0.7
0.8
0.9


BF
FF
NF
WF
FFD
1
0.8
0.6
0.4
0.2
0
0.3
0.4
0.5
0.6
0.7
0.8
0.9





We can see that in general under RM the best fit tries to fit the tasks in the first CPU while, on the other side, WF tries to use the most possible number of CPUs trying to balance the task set workload. Moreover we can see also that both FF and NF tends to assign the most possible tasks to the first CPUs and this why we need greater bandwidths for these allocation algorithms in the first CPUs. We can also see that when the utilization of the task set grows, both FF and NF needs to allocate some tasks to another CPU and this is why is some cases we can see a bandwidth greater then 0 for high values of utilization.

The same reasoning is valid also in the case of EDF of course. The difference in general is that under EDF, as we have already seen, it is simpler to schedule a task set and so the number of CPUs needed to schedule a task set is in general less than the one needed by a fixed priority algorithm.

References

[1] L. Abeni, *Hierarchical Scheduling for Components*. Slides for Component Based Software Design course at Scuola Superiore Sant' Anna, May, 2018.

[2] L. Abeni, *Supplied Time for a Periodic Server*. Slides for Component Based Software Design course at Scuola Superiore Sant' Anna, May, 2018.

[3] L. Abeni, *Again on Supplied Bound Function*. Slides for Component Based Software Design course at Scuola Superiore Sant' Anna, May, 2018.

[4] G. Buttazzo, *Hard Real-time Computing Systems: Predictable Scheduling Algorithms and Applications*. 3rd ed., Springer, 2011.

[5] C. Lu, *Real-Time Scheduling: Single Processor*. Slides at Washington University of St Louis.

[6] G. Lipari, Fixed priority *scheduling*. Slides for Real-Time Systems course at Scuola Superiore Sant' Anna, November, 2011.

[7] G. Lipari, *EDF scheduling*. Slides for Real-Time Systems course at Scuola Superiore Sant' Anna, November, 2011.

[8] G. Buttazzo, *Real-Time scheduling for multiprocessor systems*. Slides for Component Based Software Design course at Scuola Superiore Sant' Anna, April, 2015.